IT-Security (ITS) B1

DIKU, E2022

Today’s agenda

Part 1: Software vulnerabilities

Part 2: How to find and exploit vulnerabilities (guest)

Lecture plan

Week	Date	Time	Instructor	Topic	
36	05 Sep	10-12		TL	Security concepts and principles
	09 Sep	10-12		TL	Cryptographic building blocks
37	12 Sep	10-12		TL	Key establishment and certificate management
	16 Sep	10-12		CJ	User authentication, IAM
38	19 Sep	10-12		CJ	Operating systems security, web, browser and mail security
	23 Sep	10-12		CJ	IT security management and risk assessment
39	**26 Sep**	**10-12**		**TL**	**Software security - exploits and privilege escalation**
	30 Sep	10-12		TL	Malicious software
40	03 Oct	10-12		CJ	Firewalls and tunnels, security architecture
	07 Oct	10-12		CJ	Cloud and IoT security
41	10 Oct	10-12		TL	Intrusion detection and network attacks
	14 Oct	10-12		TL	Forensics
42					Fall Vacation - No lectures
43	24 Oct	10-12		CJ	Privacy and GDPR
	28 Oct	10-12		CJ	Privacy engineering
44	31 Oct	10-11		Guest	Special topic
		11-12		TL,CJ	Exam Q/A

https://github.com/diku-its/its-e2022/blob/main/lectureplan2022.md

Vulnerabilities

Vulnerabilities defined

Software contains **bugs**

A **vulnerability** is a bug that can be exploited by an attacker

An **exploit** is a piece of code that takes advantage of a vulnerability

Vulnerabilities are exploited to run **malware**

(Not all bugs can be exploited)

(Not all vulnerabilities matter the same / are equally risky)

There are many kinds of vulnerabilities

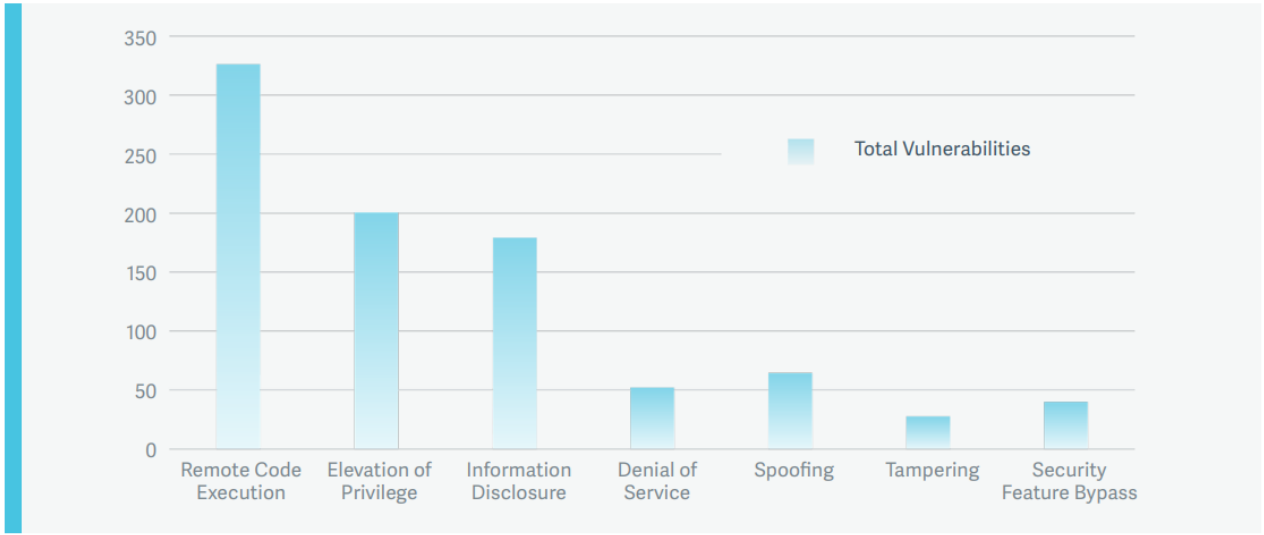
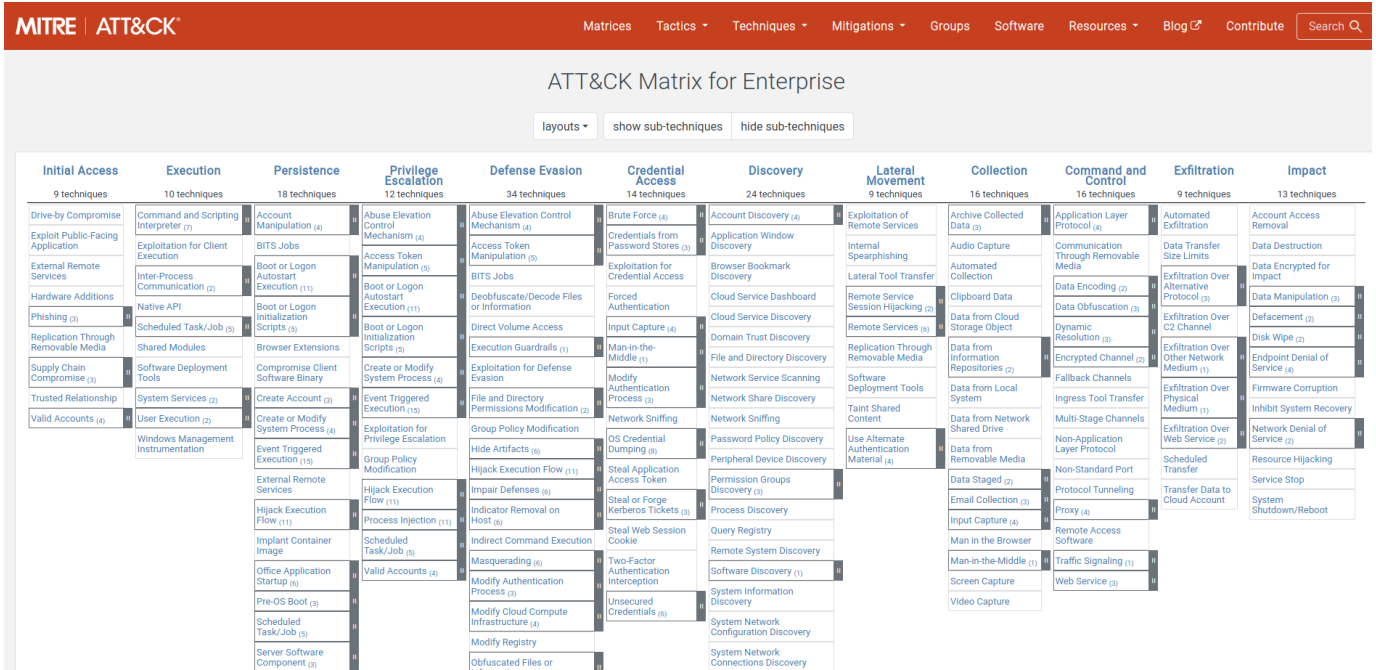


Figure 1: Breakdown of Microsoft Vulnerability Categories (2019)

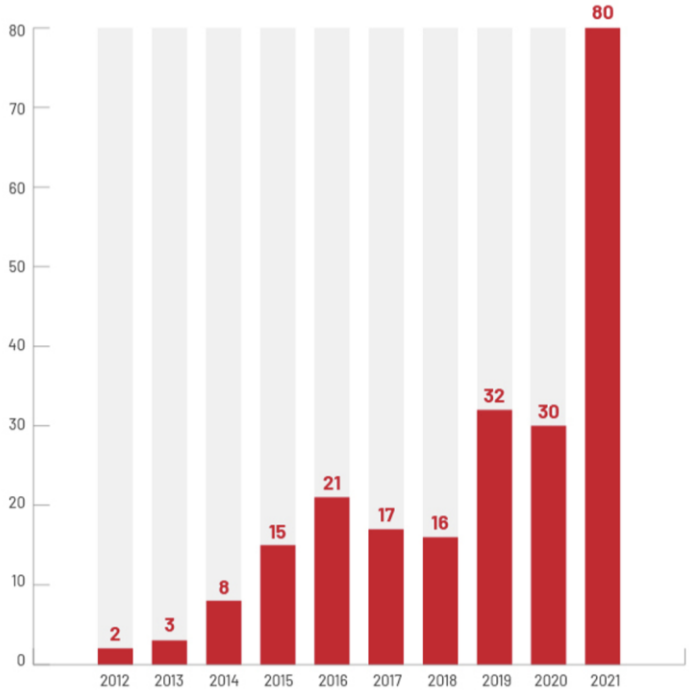
Microsoft Vulnerability Report 2020

Vulnerabilities' role in an attack



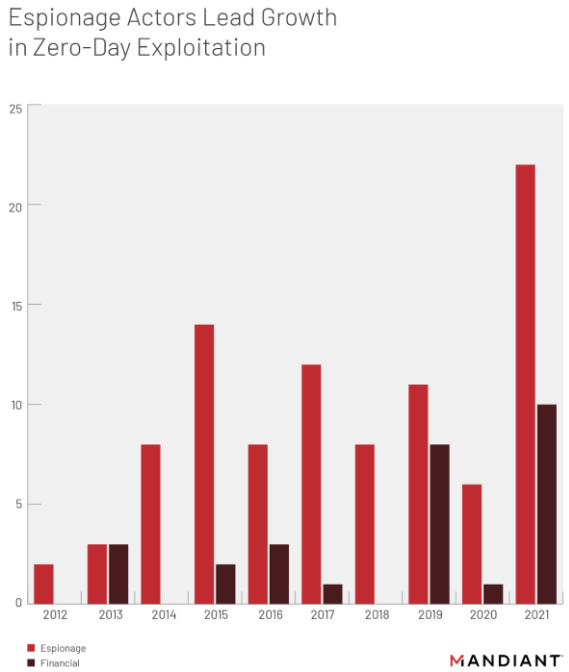
Zero-day vulnerabilities

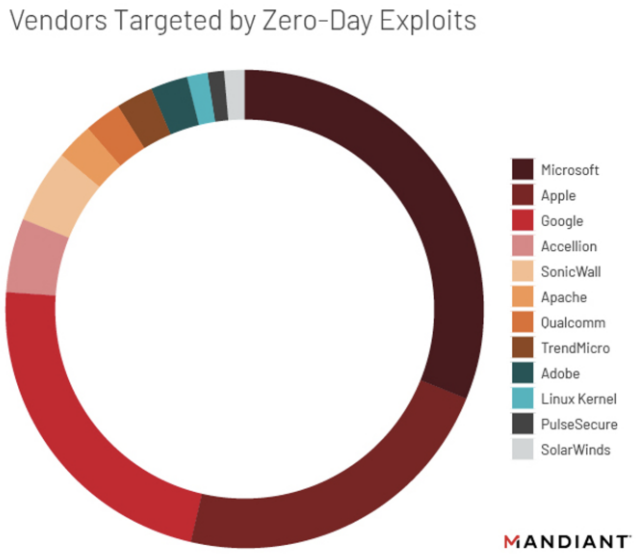
A zero-day vulnerability is a vulnerability that defenders have previously been unaware of, and for which they have had zero days to produce a fix or workaround, providing attackers the best opportunity to attack affected systems.



https://www.mandiant.com/resources/blog/zero-days-exploited-2021

Mandiant on zero-days exploited in the wild





Known Exploited Vulnerabilities (KEV) catalog

The Cybersecurity and Infrastructure Security Agency (CISA) is an agency responsible for strengthening cybersecurity across the US government

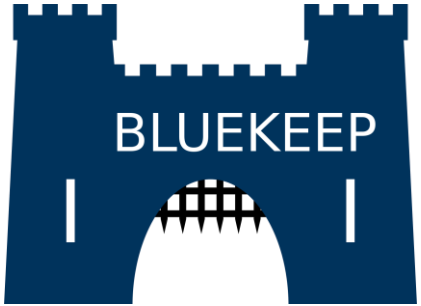
CISA maintains an authoritative source of vulnerabilities that have been exploited in the wild: the Known Exploited Vulnerability (KEV) catalog

cveID	vendorProject	dateAdded	dueDate
CVE-2022-40139	Trend Micro	15-09-2022	06-10-2022
CVE-2013-6282	Linux	15-09-2022	06-10-2022
CVE-2013-2597	Code Aurora	15-09-2022	06-10-2022
CVE-2013-2596	Linux	15-09-2022	06-10-2022
CVE-2013-2094	Linux	15-09-2022	06-10-2022
CVE-2010-2568	Microsoft	15-09-2022	06-10-2022
CVE-2022-37969	Microsoft	14-09-2022	05-10-2022
CVE-2022-32917	Apple	14-09-2022	05-10-2022
CVE-2022-3075	Google	08-09-2022	29-09-2022
CVE-2022-28958	D-Link	08-09-2022	29-09-2022
CVE-2022-27593	QNAP	08-09-2022	29-09-2022
CVE-2022-26258	D-Link	08-09-2022	29-09-2022
CVE-2020-9934	Apple	08-09-2022	29-09-2022
CVE-2018-7445	MikroTik	08-09-2022	29-09-2022
CVE-2018-6530	D-Link	08-09-2022	29-09-2022
CVE-2018-2628	Oracle	08-09-2022	29-09-2022
CVE-2018-13374	Fortinet	08-09-2022	29-09-2022
CVE-2017-5521	NETGEAR	08-09-2022	29-09-2022
CVE-2011-4723	D-Link	08-09-2022	29-09-2022

This is a vulnerability

BlueKeep (CVE-2019-0708) is a vulnerability that was discovered in Microsoft's Remote Desktop Protocol (RDP) implementation, which allows for the possibility of remote code execution.

First reported in May 2019, Microsoft issued a security patch (including an out-of-band update for several versions of Windows that have reached their end-of-life, such as Windows XP) on 14 May 2019. On 6 September 2019, a Metasploit exploit of the wormable BlueKeep security vulnerability was publicly released.



Kevin Beaumont ✓
@GossiTheDog

CVE-2019-0708 RDP vulnerability megathread, aka BlueKeep.

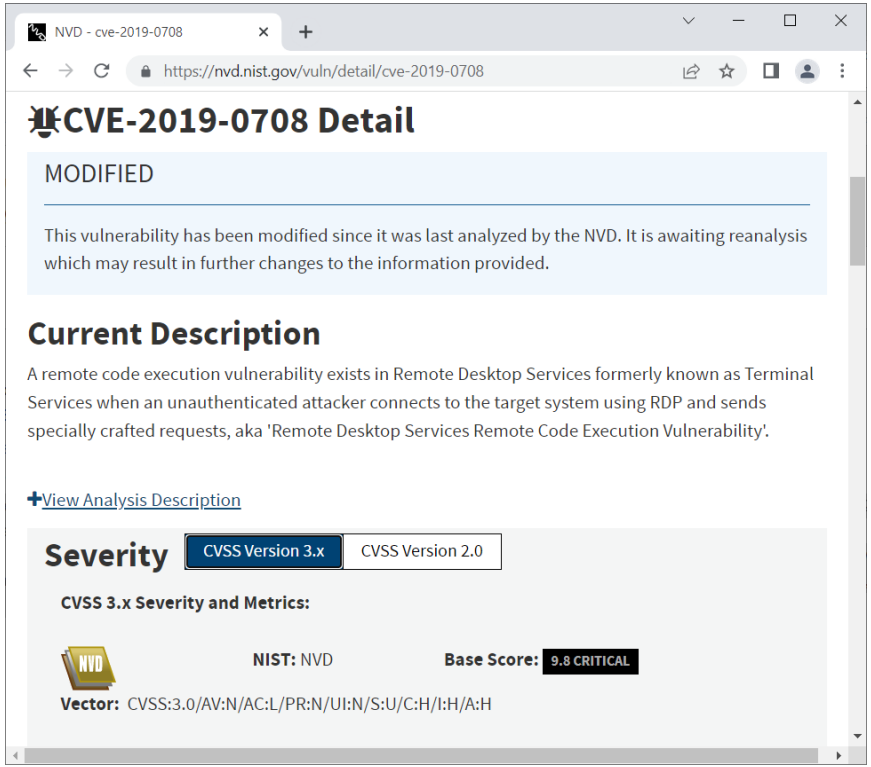
Going to nickname it BlueKeep as it's about as secure as the Red Keep in Game of Thrones, and often leads to a blue screen of death when exploited.

12.47 AM · 15. maj 2019 · Twitter for iPhone

This is a vulnerability

BlueKeep (CVE-2019-0708) is a vulnerability that was discovered in Microsoft's Remote Desktop Protocol (RDP) implementation, which allows for the possibility of remote code execution.

First reported in May 2019, Microsoft issued a security patch (including an out-of-band update for several versions of Windows that have reached their end-of-life, such as Windows XP) on 14 May 2019. On 6 September 2019, a Metasploit exploit of the wormable BlueKeep security vulnerability was publicly released.



CVE and CVSS

The Common Vulnerabilities and Exposures (CVE) system provides a reference-method for publicly known vulnerabilities.

The Common Vulnerability Scoring System (CVSS) is a free and open industry standard for assessing the severity of vulnerabilities.

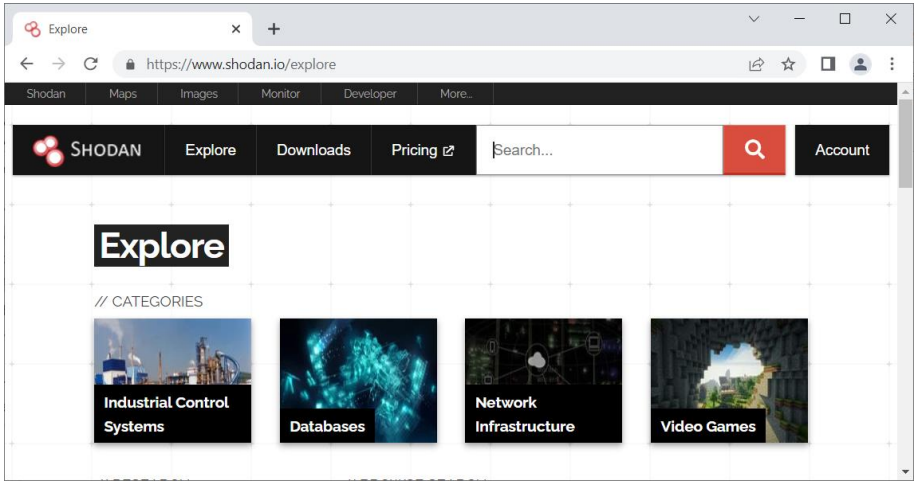
CVSS v3.0 Severity and Metrics:
Base Score: 9.8 CRITICAL
Vector: AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
Impact Score: 5.9
Exploitability Score: 3.9

Attack Vector (AV): Network
Attack Complexity (AC): Low
Privileges Required (PR): None
User Interaction (UI): None
Scope (S): Unchanged
Confidentiality (C): High
Integrity (I): High
Availability (A): High

Shodan and BleeKeep

Shodan is a search engine that lets users search for various types of servers (webcams, routers, servers, etc.) connected to the internet using a variety of filters.

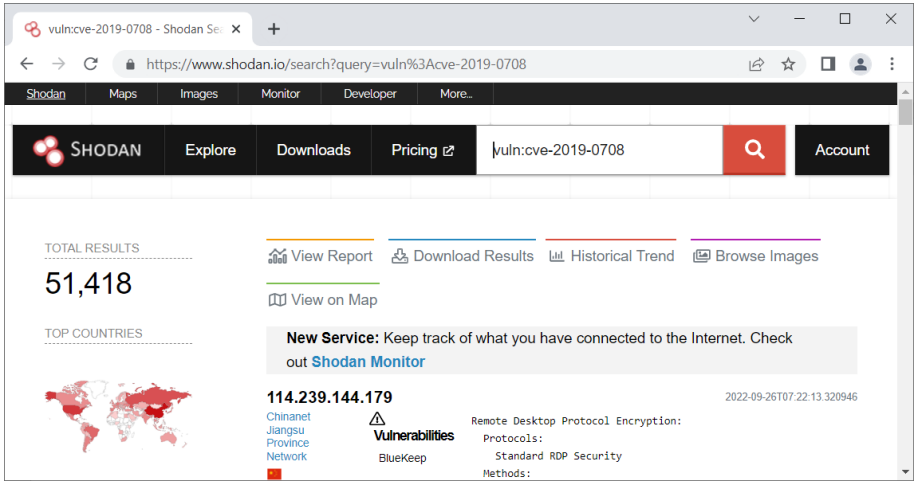
This can be information about the server software, what options the service supports, a welcome message or anything else that the client can find out before interacting with the server.



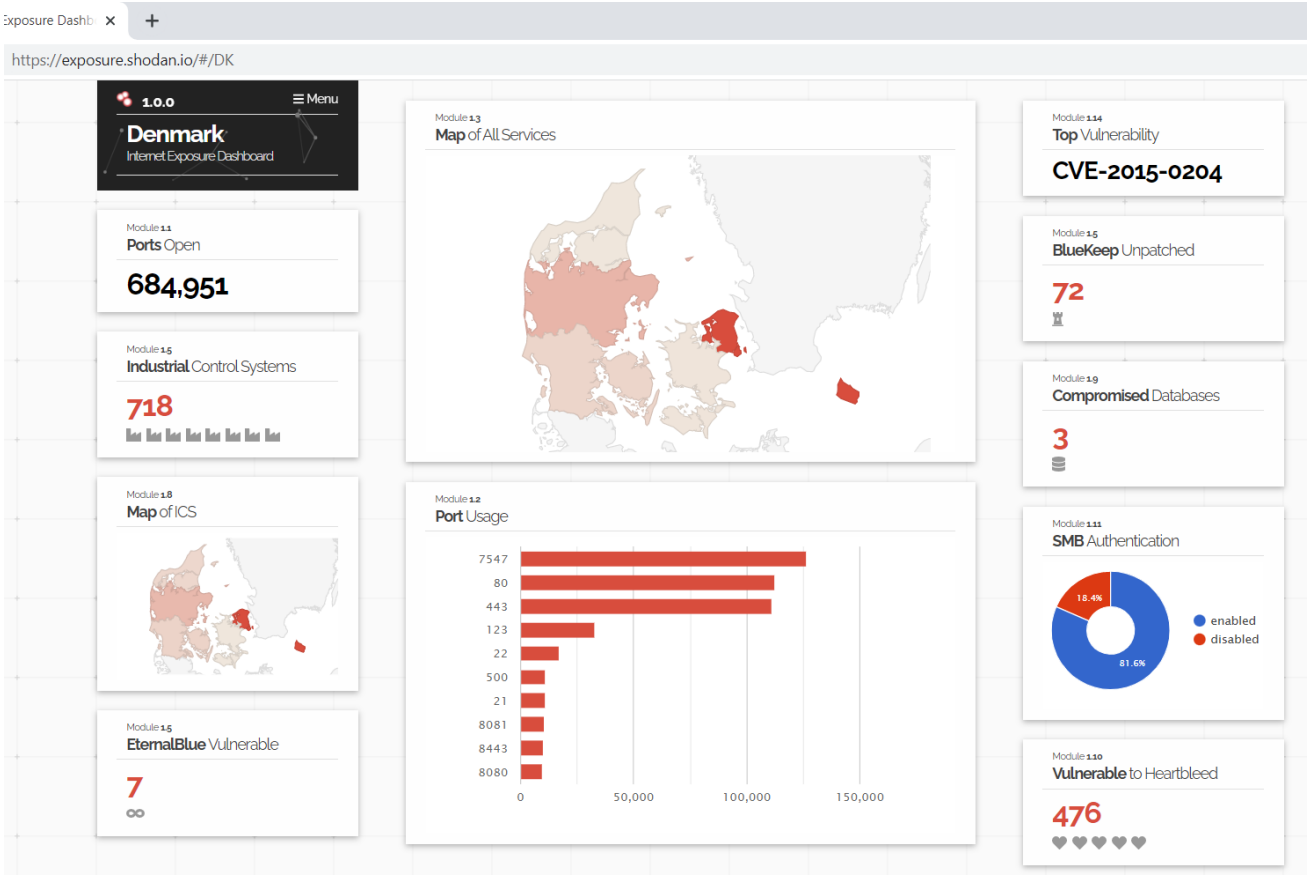
Shodan and BleeKeep

Shodan is a search engine that lets users search for various types of servers (webcams, routers, servers, etc.) connected to the internet using a variety of filters.

This can be information about the server software, what options the service supports, a welcome message or anything else that the client can find out before interacting with the server.



Shodan and “Denmark”



Wrap-up

Lecture plan

Week	Date	Time	Instructor	Topic	
36	05 Sep	10-12		TL	Security concepts and principles
	09 Sep	10-12		TL	Cryptographic building blocks
37	**12 Sep**	**10-12**		**TL**	**Key establishment and certificate management**
	16 Sep	10-12		CJ	User authentication, IAM
38	19 Sep	10-12		CJ	Operating systems security, web, browser and mail security
	23 Sep	10-12		CJ	IT security management and risk assessment
39	26 Sep	10-12		TL	Software security - exploits and privilege escalation
	30 Sep	10-12		TL	Malicious software
40	03 Oct	10-12		CJ	Firewalls and tunnels, security architecture
	07 Oct	10-12		CJ	Cloud and IoT security
41	10 Oct	10-12		TL	Intrusion detection and network attacks
	14 Oct	10-12		TL	Forensics
42					Fall Vacation - No lectures
43	24 Oct	10-12		CJ	Privacy and GDPR
	28 Oct	10-12		CJ	Privacy engineering
44	31 Oct	10-11		Guest	Special topic
		11-12		TL,CJ	Exam Q/A

https://github.com/diku-its/its-e2022/blob/main/lectureplan2022.md

IT-Security (ITS) B1

DIKU, E2022

Types of vulnerabilities, include:

Format string

Overflow

Over-read

Load order

Use-after-free

Dangling pointers

Code injection

Command injection

Race conditions

Typos, and more

Where’s the bug?

```c
#include <stdio.h>

int main () {
    int i;
    printf("Enter a value: ");
    scanf("%d", &i);

    if (i < 0)
        goto fail;
    if (i > 100)
        goto fail;
        goto fail;
    if (i%2 == 0)
        goto fail;

    return;

fail:
    printf("Fail\n");
    return;
}
```

```
$ ./a.out
Enter a value: 2
Fail

$ ./a.out
Enter a value: 3
Fail
```

```c
#include <stdio.h>

int main () {
    int i;
    printf("Enter a value: ");
    scanf("%d", &i);

    if (i < 0)
        goto fail;
    if (i > 100)
        goto fail;
        //goto fail;
    if (i%2 == 0)
        goto fail;

    return;

fail:
    printf("Fail\n");
    return;
}
```

```
$ ./a.out
Enter a value: 2
Fail

$ ./a.out
Enter a value: 3
Fail
```

Apple iOS Goto Fail

```
1  static OSStatus
2  SSLVerifySignedServerKeyExchange(SSLContext *ctx, bool isRsa, SSLBuffer signedParams,
3                                   uint8_t *signature, UInt16 signatureLen)
4  {
5      OSStatus        err;
6      ...
7
8      if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
9          goto fail;
10     if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
11         goto fail;
12         goto fail;
13     if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
14         goto fail;
15     ...
16
17 fail:
18     SSLFreeBuffer(&signedHashes);
19     SSLFreeBuffer(&hashCtx);
20     return err;
21 }
```

```c
#include <stdio.h>
#include <string.h>

int main () {

   char buf[20] = "http://www.diku.dk";
   char shh[30] = "mumstheword";
   char out[64];
   int chars;

   printf("Buffer contents: %s\n", buf);

   printf("Chars to copy: ");
   scanf("%d", &chars);

   memcpy(out, buf, chars);

   printf("Copied: ");
   fwrite(out, chars, 1, stdout);
   printf("\n");

   return(0);
}
```

```
$ ./a.out
Buffer contents: http://www.diku.dk
Chars to copy: 12
Copied: http://www.d

$ ./a.out
Buffer contents: http://www.diku.dk
Chars to copy: 50
Copied: http://www.diku.dk¨0L H¨mumstheword
```

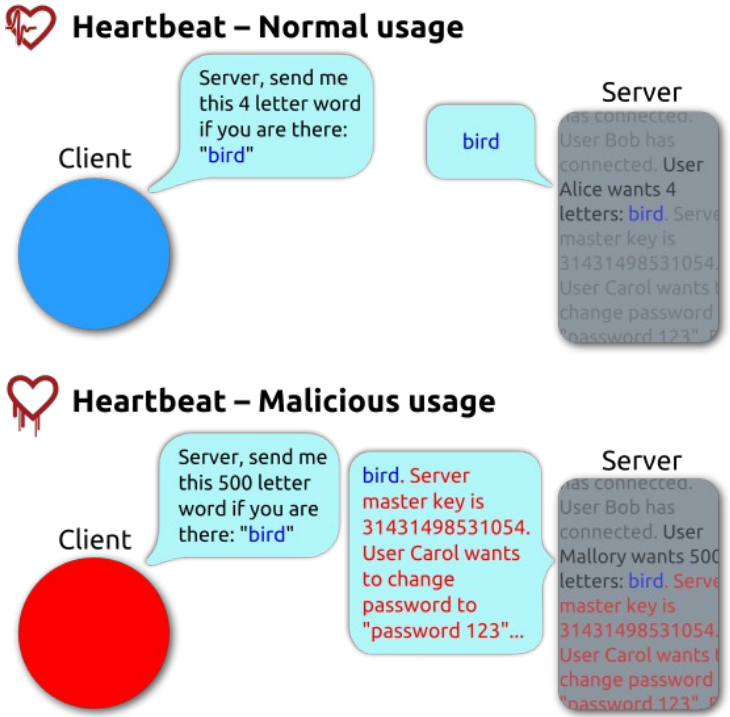
```c
#include <stdio.h>
#include <string.h>

int main () {

   char buf[20] = "http://www.diku.dk";
   char shh[30] = "mumstheword";
   char out[64];
   int chars;

   printf("Buffer contents: %s\n", buf);

   printf("Chars to copy: ");
   scanf("%d", &chars);
   if (chars > sizeof(buf)) chars = sizeof(buf);
   memcpy(out, buf, chars);

   printf("Copied: ");
   fwrite(out, chars, 1, stdout);
   printf("\n");
```

```
$ ./a.out
Buffer contents: http://www.diku.dk
Chars to copy: 12
Copied: http://www.d

$ ./a.out
Buffer contents: http://www.diku.dk
Chars to copy: 50
Copied: http://www.diku.dk¨0L H¨mumstheword
```

The HeartBleed Bug

Heartbleed was a security bug in the OpenSSL cryptography library, which is a widely used implementation of the Transport Layer Security (TLS) protocol. It was introduced into the software in 2012 and publicly disclosed in April 2014.

Heartbleed could be exploited regardless of whether the vulnerable OpenSSL instance is running as a TLS server or client. It resulted from improper input validation (due to a missing bounds check) in the implementation of the TLS heartbeat extension.



```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv)
{

   printf("Current time: ");
   fflush(stdout);
   system("date");
   return 0;

}
```

```
$ ./a.out
Current time: Mon Sep  26 10:35:47 CEST 2022

$ export PATH=`pwd`:$PATH
$ echo -e '#!/bin/sh\necho "Hello"' > date
$ chmod 700 date

$ ./a.out
Current time: Hello
```

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv)
{

   printf("Current time: ");
   fflush(stdout);
   system("/bin/date");
   return 0;

}
```

```
$ ./a.out
Current time: Mon Sep  26 10:35:47 CEST 2022

$ export PATH=`pwd`:$PATH
$ echo -e '#!/bin/sh\necho "Hello"' > date
$ chmod 700 date

$ ./a.out
Current time: Hello
```

Real-world example: PlugX

PlugX drops

- A legitimate NVIDIA file (NvSmart.exe)
- A malicious DLL (NvSmartMax.dll)

Normally, NvSmart.exe would load a legitimate NvSmartMax.dll

But if a (malicious) version the DLL file is located in the same directory, this will load instead

```
#!/usr/bin/perl

open(FH, $ARGV[0]);

while(<FH>)
{
  print $_;
}

close(FH);
```

```
$ ./code.pl code.pl
#!/usr/bin/perl

open(FH, $ARGV[0]);

while(<FH>)
{
  print $_;
}

close(FH);

$ ./code.pl 'ls -l code.pl|'
-rwx------ 1 user user 79 Sep  26 10:41 code.pl
```

```
#!/usr/bin/perl

open(FH, "< ".$ARGV[0]); #force read open with '<'

while(<FH>)
{
  print $_;
}

close(FH);
```

```
$ ./code.pl code.pl
#!/usr/bin/perl

open(FH, $ARGV[0]);

while(<FH>)
{
  print $_;
}

close(FH);

$ ./code.pl 'ls -l code.pl|'
-rwx------ 1 user user 79 Sep  26 10:41 code.pl
```

Explanation

According to the Perl documentation

> If filename ends with a "|", filename is interpreted as a command which pipes output

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char **argv)
{

   char buffer[64];
   strncpy(buffer, argv[1], sizeof(buffer));

   printf("You entered: ");
   printf(buffer);
   printf("\n");
}
```

```
$ ./a.out A
You entered: A

$ ./a.out %s
You entered: You entered:

$ ./a.out %x
You entered: 510a2000

$ ./a.out %x%x
You entered: 437a00041569e0
```

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char **argv)
{

   char buffer[64];
   strncpy(buffer, argv[1], sizeof(buffer));

   printf("You entered: ");
   printf("%s", buffer);
   printf("\n");
}
```

```
$ ./a.out A
You entered: A

$ ./a.out %s
You entered: You entered:

$ ./a.out %x
You entered: 510a2000

$ ./a.out %x%x
You entered: 437a00041569e0
```

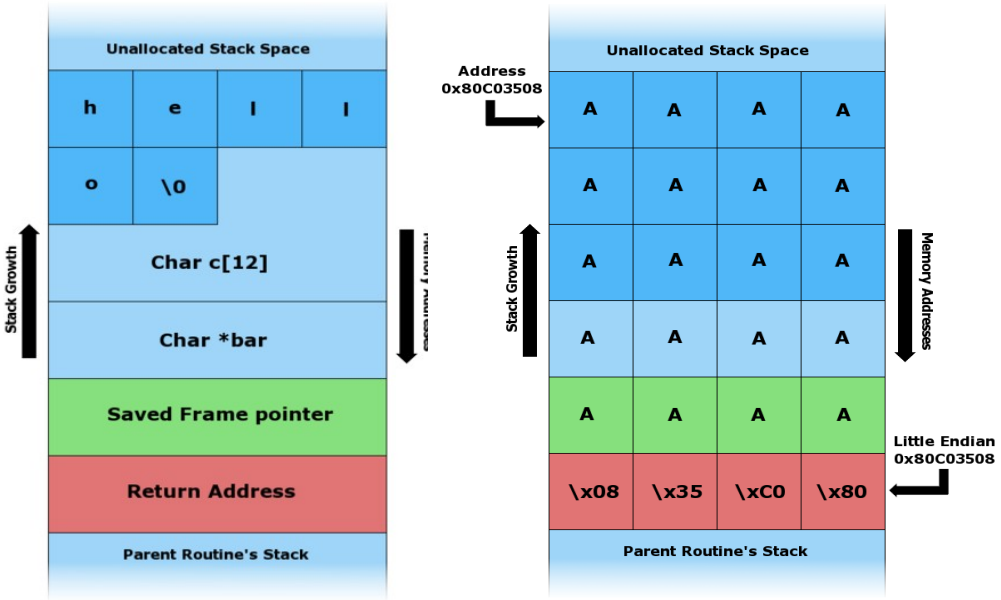
```
#include <string.h>

void foo (char *bar)
{
  char  c[12];
  strcpy(c, bar);
}

int main (int argc, char **argv)
{
  foo(argv[1]);
}
```

```
$ ./6.out A

$ ./6.out AAAAAAAAAAAAAA

$ ./6.out AAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
```

```
#include <string.h>

void foo (char *bar)
{
  char  c[12];
  strncpy(c, bar, sizeof(c));
}

int main (int argc, char **argv)
{
  foo(argv[1]);
}
```

```
$ ./6.out A

$ ./6.out AAAAAAAAAAAAAA

$ ./6.out AAAAAAAAAAAAAAAAAAAAAAAA
Segmentation fault
```

```
#include <string.h>

void foo (char *bar)
{
  char  c[12];
  strcpy(c, bar);
}

int main (int argc, char **argv)
{
  foo(argv[1]);
}
```



Some countermeasures

Stack canaries

- Check stack not altered when function returns

Data execution prevention (DEP)

- Prevent the execution of data on the stack or heap

Address space layout randomization (ASLR)

- Rearrange memory positions to make successful exploitation more difficult